

**Software Design**  
**CSE 335, Spring 2006**

**Object-oriented programming: Generalization  
Hierachies**

Stephen Wagner

Michigan State University

# Generalization

- **Defn:** relationship between a class and one or more refined versions of it.
- Benefits:
  - Powerful abstraction for sharing similarities among classes while preserving differences
  - Inheritance is helpful as a tool for reusing code
- Not a synonym for inheritance!
  - Can inherit from a class that is not a generalization
  - Can define generalizations that cannot (easily) be implemented with inheritance

## Example

- Consider STL list implementation

```
template <typename T>
class list {
public:
    bool    empty() const;
    void    push_front( const T& );
    void    push_back( const T& );
    void    pop_front();
    void    pop_back();
};
```

- A stack can be implemented with a list

```
template <typename T>
class list {
public:
    bool    empty() const;
    void    push_front( const T& );
    void    push_back( const T& );
    void    pop_front();
    void    pop_back();
};

class stack : public list {
public:
    ...
}
```

Is this a good idea?

## A Stack *is not* a List

- A stack has `push_back` and `pop_back`
- stack inherits all of the list operations

```
stack Foo;
```

```
Foo.pop_front(); // the compiler allows this  
                // does not make sense
```

# Liskov Substitution Principle (LSP)

- Design Principle: a derived class should be logical, consistent extension of its base class
- Liskov Substitution Principle (LSP):
  - *An instance of a class should function as an instance of its base class*
- LSP helps one to use inheritance to implement generalization/specialization (is-a relationship)

# Stacks and Lists

- Can a list inherit from a stack?

## An Example

- Consider an HTML like formatting language that supports paragraphs, lists, and ordered lists. A paragraph contains a sequence of words. A list contains a sequence of paragraphs or lists. What sort of classes do we need?

## An Example

- Consider an HTML like formatting language that supports paragraphs, lists, and ordered lists. A paragraph contains a sequence of words. A list contains a sequence of paragraphs or lists. What sort of classes do we need?
- A table consists of a sequence of rows. Each row contains a sequence of cells, and each cell contains formatted text. How do we implement this?

# UML

- In large systems it is important to understand the relationships between classes
- *Class model* is a graphical depiction of the classes and their relationships
- Unified Modelling Language (UML)

# Basic UML

- Describing a class

<b>Class Name</b>
attribute : type : default
operation(arglist) : type

- Attributes and operations are optional

# Generalization in UML

